Ciências ULisboa | Informática

di.ciencias.ulisboa.pt

/difcul

Segurança e Confiabilidade

2023/2024

Trabalho 1 – 1ª fase

1 Objetivos

A parte prática da disciplina de Segurança e Confiabilidade pretende familiarizar os alunos com alguns dos problemas envolvidos na programação de aplicações distribuídas seguras, nomeadamente a gestão de chaves criptográficas, a geração de sínteses seguras, atestação remota, cifras e assinaturas digitais, e a utilização de canais seguros. O primeiro trabalho a desenvolver na disciplina será realizado utilizando a linguagem de programação Java e a API de segurança do Java, e é composto por duas fases. A primeira fase do projeto está enunciada neste documento, e tem como objetivo fundamental a construção de uma aplicação distribuída, focando-se essencialmente nas funcionalidades da aplicação. Na fase 2 do projeto, iremos dedicar os nossos esforços para garantir que os todos os processos – transmissão de mensagens, salvaguarda de dados, ou execução de software – são seguros e confiáveis. O enunciado da segunda fase será oportunamente divulgado.

O trabalho envolve a implementação de um sistema de IoT com arquitetura cliente-servidor mostrada na Fig.1. Um **cliente** IoT é um programa – chamemos-lhe `IoTDevice` – que representa um dispositivo de sensorização. Este programa é responsável por enviar dados sensoriais para o servidor. Este programa é controlado por um utilizador apenas, e tem um identificador único (exº utilizador `alan` controla os dispositivos `alan:3` e `alan:1`).

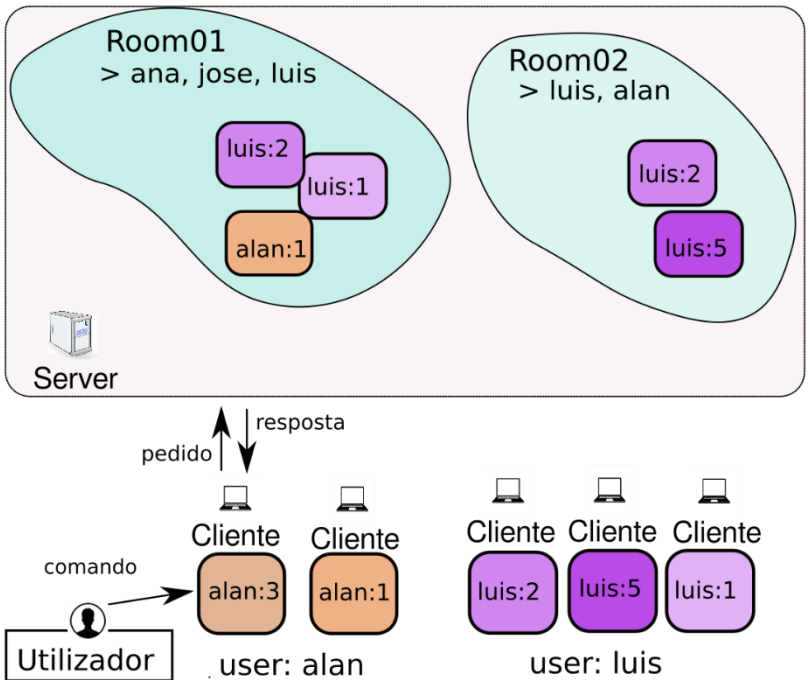

Fig.1. Diagrama da arquitetura do sistema IoT, com um Servidor e múltiplos clientes.

Para um IoTDevice publicar dados no servidor, o utilizador deverá autenticar-se no servidor primeiramente. Para poder aceder aos dados do servidor, existe o conceito de Domínio. Os dispositivos podem fazer parte de um ou mais domínios de trabalho (cf Fig.1.: o dispositivo luis:2 está registado no domínio Room01 e Room02). Desta forma, um utilizador com acesso de leitura a um dado domínio pode aceder aos dados gerados por qualquer dispositivo dentro desse domínio (ex.: o utilizador *luis* está registado no domínio Room01 e Room02, e, portanto, poderá ler dados de 4 dispositivos, nomeadamente: luis:2, luis:1, alan:1, luis:5. O utilizador jose está registado no domínio Room01, logo apenas poder ler dados de luis:2, luis:1 e alan:1. Já alan pode ler dados de luis:2 e luis:5.

A aplicação IoTDevice têm duas grandes funcionalidades: 1) Enviar (dois tipos de) dados para o servidor: imagens e valores de temperatura; 2) Aceder a dados de temperatura e/ou imagens de um dado dispositivo. Para fazer uso de todas estas funcionalidades, a aplicação IoTDevice deverá implementar uma linha de comandos simples, para interação com o utilizador.

A aplicação **servidor,** nomeada *IoTServer,* é um programa que permite a ligação com vários clientes em simultâneo, mantém informação sobre os dispositivos, domínios e utilizadores registados, autentica e identifica utilizadores, e vai colecionando e partilhando informação já recebida dos vários clientes, de forma organizada e persistente.

2 Arquitetura do Sistema

O trabalho consiste no desenvolvimento de dois programas:

- A aplicação servidor IoTServer (espera ligações TCP)
- A aplicação cliente IoTDevice (acede ao servidor via TCP)

O servidor deverá poder executar numa máquina e permitir ligar um número alargado de clientes a serem executados simultaneamente em máquinas distintas. A execução das aplicações deve ser feita da seguinte forma:

1. Servidor:

   IoTServer [port]

   - [port] identifica o porto (TCP) para aceitar ligações de clientes. Por omissão, o servidor deve usar o porto 12345 e aceitar ligações em qualquer interface.

2. Cliente:

   IoTDevice <serverAddress> <dev-id> <user-id>

   Em que:

   - <serverAddress> identifica o servidor. O formato de serverAddress é o seguinte: <IP/hostname>[:Port]. O endereço IP/hostname do servidor é obrigatório e o porto é opcional. Por omissão, o cliente deve ligar-se ao porto 12345 do servidor.
   - <dev-id> - número inteiro que identifica o dispositivo.
   - <user-id> – string que identifica o (endereço de email do) utilizador local.

3 Funcionalidades

Cliente

A aplicação cliente deve ter o seguinte comportamento sequencial:

1. Começa por pedir a senha, e envia o *user-id e a senha* ao servidor, para tentar autenticar o utilizador.
2. Um de três cenários podem ocorrer:
    a. O cliente não consegue autenticar no servidor – este receberá uma mensagem do tipo WRONG-PWD. O utilizador terá oportunidade de voltar a tentar outra password.
    b. O cliente não existe registado no servidor – este será registado no servidor, e recebe uma mensagem do tipo OK-NEW-USER.
    c. O cliente existe e a password está correta – o cliente recebe uma mensagem do tipo OK-USER.
3. O cliente envia seu `<device-id>` ao servidor.
4. Dois cenários podem ocorrer:
    a. O cliente recebe uma mensagem NOK-DEVID, no caso de já existir outro `IoTDevice` aberto e autenticado com o mesmo par (`<user-id>`,`<dev-id>`). Nesse caso, o utilizador deverá ter oportunidade de inserir um novo device ID. Voltar ao passo 3.
    b. O cliente recebe uma mensagem OK-DEVID, e o processo segue em frente.
5. O cliente envia o nome e o tamanho do ficheiro executável `IoTDevice`, para ser atestado remotamente.
6. Dois cenários podem ocorrer:
    a. O servidor não valida o programa cliente – o cliente recebe a mensagem do tipo NOK-TESTED, a ligação é terminada, e o cliente termina assinalando o erro respetivo.
    b. O servidor valida o programa cliente – o cliente recebe a mensagem do tipo OK-TESTED, e o processo segue em frente.
7. Em seguida, a `IoTDevice` deverá apresentar um menu com os comandos disponíveis ao utilizador, nomeadamente o seguinte texto:

- **CREATE <dm>** # Criar domínio - utilizador é Owner
- **ADD <user1> <dm>** # Adicionar utilizador <user1> ao domínio <dm>
- **RD <dm>** # **R**egistar o **D**ispositivo atual no domínio <dm>
- **ET <float>** # **E**nviar valor <float> de **T**emperatura para o servidor.
- **EI <filename.jpg>** # **E**nviar **I**magem <filename.jpg> para o servidor.
- **RT <dm>** # **R**eceber as últimas medições de **T**emperatura de cada dispositivo do domínio <dm>, desde que o utilizador tenha permissões.
- **RI <user-id>:<dev_id>** # **R**eceber o ficheiro **I**magem do dispositivo <user-id>:<dev_id> do servidor, desde que o utilizador tenha permissões.

8. A aplicação cliente deverá implementar uma linha de comandos que permita a utilização dos comandos apresentados no passo 7.
9. O utilizador introduz um dos comandos disponíveis, e um pedido respetivo é enviado ao servidor. O cliente recebe a resposta do servidor e imprime essa informação no terminal.
10. Enquanto o utilizador não pressionar CTRL+C, o programa deverá voltar ao passo 9. Caso contrário, termina.

Comandos Disponíveis

O comando CREATE <dm> deverá criar um novo domínio no servidor. Caso o servidor aceite o pedido, o cliente deve receber uma mensagem do tipo OK. Caso o domínio já exista, deverá receber uma mensagem do tipo NOK.

Exemplo

Comando: **CREATE Room01**

Resposta: OK

O comando **ADD <user1> <dm>** deve permitir adicionar o utilizador <user1> ao domínio <dm>. Caso o servidor aceite o pedido, o cliente deverá receber uma mensagem do tipo OK.

Se o Utilizador que faz o pedido não é o criador do domínio, o domínio não existe, ou o utilizador não existe, então o cliente deverá receber uma mensagem do tipo NOPERM, NODM, ou NOUSER, respetivamente.

Exemplo:

Comando: **ADD alan Room01**

Resposta: OK

Comando: **ADD alan Room02**

Resposta: NODM # esse domínio não existe

Comando: **ADD alan Room03**

Resposta: NOPERM # sem permissões

O comando **RD <dm>** permite registar o dispositivo atual no domínio <dm>. Caso o servidor aceite a informação, o cliente deve receber uma mensagem do tipo OK. Caso o utilizador atual não pertença ao domínio <dm>, ou o domínio não exista o cliente deverá receber uma mensagem do tipo NOPERM ou NODM, respetivamente.

Exemplo:

Comando: **RD Room01**

Resposta: OK

Comando: **RD Room02**

Resposta: NODM # esse domínio não existe

Comando: **RD Room03**

Resposta: NOPERM # sem permissões

O comando **ET <float>** deve enviar o valor <float> da temperatura, como um número real. Caso o servidor aceite a informação, o cliente deve receber uma mensagem do tipo OK. Caso contrário recebe, uma mensagem do tipo NOK.

Exemplo:

Comando: ET 27.31

Resposta: OK

Comando: ET batata

Resposta: NOK

O comando **EI <filename.jpg>** deverá abrir o ficheiro do cliente com o nome filename.jpg, e enviar uma cópia deste ao Servidor. Caso o servidor aceite a informação, o cliente deve receber uma mensagem do tipo OK. Caso contrário recebe, uma mensagem do tipo NOK.

Exemplo:

Comando: EI frame01.jpg

Resposta: OK

O comando **RT <dm>** permite ao programa IoTDevice obter um ficheiro de texto com os dados de temperatura gravado no servidor, de todos os dispositivos dentro do domínio <dm>. Assim o cliente deverá receber uma mensagem OK, depois um número LONG com o tamanho do ficheiro, e em seguida o conteúdo do ficheiro. Caso o pedido seja inválido, o cliente poderá receber uma mensagem NODATA, NODM ou NOPERM por parte do servidor.

Exemplo:

Comando: **RT Room01**

Resposta: OK, 45 (long), seguido de *45 bytes de dados.*

Comando: **RT Room02**

Resposta: NODM # esse domínio não existe

Comando: **RT Room03**

Resposta: NOPERM # sem permissões de leitura

O comando **RI <user_id>:<dev_id>** permite ao programa IoTDevice obter a imagem do dispositivo <user_id>:<dev_id>. Assim o cliente começa por receber uma mensagem OK, depois um valor LONG com o tamanho do ficheiro, e de seguida recebe o ficheiro propriamente dito. Caso o pedido seja inválido, o cliente poderá receber uma mensagem NODATA, NOID ou NOPERM por parte do servidor.

Exemplo:

Comando: **RI luis:1**

Resposta: OK, *4500 (long), seguido de 4500 Bytes de dados.*

Comando: **RI luis:2**

Resposta: NODATA # esse device id não publicou dados

Comando: **RI luis:3**

Resposta: NOID # esse device id não existe

Comando: **RI luis:4**

Resposta: NOPERM # sem permissões de leitura

Servidor

O servidor mantém um **ficheiro de texto** com os **utilizadores** do sistema e as suas respetivas senhas. Este ficheiro deverá ser utilizado quando o cliente se tenta autenticar com o servidor.

O servidor mantém um **ficheiro de texto** com os **domínios** existentes, nomeadamente o seu nome e dispositivos que lhe estão associados. Esse ficheiro deverá também manter a lista de utilizadores que têm permissões de leitura de cada domínio. Este ficheiro deverá ser utilizado quando o cliente se tenta ler dados do servidor.

Caso um utilizador se tente autenticar, e não esteja registado no servidor, o Servidor fará o registo do novo utilizador, adicionando o *user-id* e a respetiva *senha* ao ficheiro de utilizadores do servidor. Devem também ser inicializadas as estruturas de dados que mantêm informações relativas a cada utilizador.

Após a autenticação, o servidor ao receber o tamanho do programa `IoTDevice` por parte do cliente, o servidor deve validar este número com a sua informação local. O servidor mantém um **ficheiro de texto** com o nome da aplicação e o seu tamanho. Para os dados sensoriais, o servidor apenas guarda os últimos dados enviados por cada dispositivo: um valor de temperatura e uma imagem.

4 Entrega

O prazo de entrega é no dia **27 de março, até às 23:59 horas**.

O código desta primeira fase do trabalho deve ser entregue de acordo com as seguintes regras:

- Para entregar o trabalho, é necessário criar um **ficheiro zip** com o nome **SegC-grupoXX-proj1-fase1.zip**, onde **XX** é o número do grupo, contendo:
  - ✓ o código do trabalho;
  - ✓ os ficheiros jar (cliente e servidor) para execução do projeto;
  - ✓ um readme (txt) indicando **como compilar** e **como executar** o projeto, indicando também as limitações do trabalho.
- O ficheiro zip é submetido através da atividade disponibilizada para esse efeito na página da disciplina no Moodle. Apenas um dos elementos do grupo deve submeter. Se existirem várias submissões do mesmo grupo, será considerada a mais recente.
- Não serão aceites trabalhos enviados por email. Se não se verificar algum destes requisitos o trabalho é considerado não entregue.

5 Autoavaliação de contribuições

Cada aluno tem de preencher no Moodle um formulário de autoavaliação das contribuições individuais de cada elemento do grupo para o projeto. Por exemplo, se todos os elementos colaboraram de forma idêntica, bastará que todos indiquem que cada um contribuiu 33%. Aplicam-se as seguintes regras e penalizações:

- Alunos que não preencham o formulário **sofrem uma penalização na nota de 10%.**
- Caso existam assimetrias significativas entre as respostas de cada elemento do grupo, o grupo poderá ser chamado para as explicar.
- Se as contribuições individuais forem diferentes, isso será refletido na nota de cada elemento do grupo, levando à atribuição de notas individuais diferentes.

O prazo de preenchimento é o mesmo que o da entrega do projeto (**27 de março, até às 23:59 horas**).